VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A Computer Graphics & Visualization Mini Project Report
on

"2D HELICOPTER GAME"

Submitted in Partial fulfillment of the Requirements for VI Semester of the Degree of

Bachelor of Engineering
In
Computer Science & Engineering
By

DADIREDDY SAI KUMAR REDDY
(1CR19CS035)

DARSHAN R
(1CR19CS037)

Under the Guidance of

Mr. Shivraj B
Asst Professor, Dept. of CSE
and
Dr. Kiran Babu T S
Asst Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Computer Graphics & Visualization project work entitled **“2D HELICOPTER GAME”** has been carried out by **Dadireddy Sai Kumar Reddy (1CR19CS035)** and **Darshan R (1CR19CS037)** bonafide students of CMR Institute of Technology in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2021-2022**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. This CG project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

-----------------------	**----------------------**
Signature of Guide	**Signature of HOD**
Mr. Shivraj B **Asst Professor** **Dept. of CSE, CMRIT**	**Dr. Shreekanth M Prabhu** **Professor & Head** **Dept. of CSE, CMRIT**

External Viva

Name of the examiners	Signature with date
1.	
2	

ABSTRACT

Computer Graphics plays major role in the development of games. This is one of the most interesting game. It's simple, easy to play, easy to understand and very easy to code, providing you guys with 2D Helicopter Game computer graphics with OpenGL. As the name suggests, the game is all about egg. OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects. Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are: Main GL, OpenGL Utility Library (GLU) and OpenGL Utility Toolkit (GLUT).

A 2d graphics based game helicopter is a great start for a student who starts learning Computer Graphics & visualization. The development of the game has large scope to learn computer graphics from scratch. We used OpenGL utility toolkit to implement the algorithm, written in C++ language. A 2D Helicopter game is computer graphics project based on the OpenGL functions The game consists of a helicopter, blocks designed using rectf() function of OpenGL. Instructions for the game here are if the helicopter hits the blocks or the walls that are moving horizontally towards left, we need to escape from the blocks and move forward to score in the game ,if we either hit the incoming blocks or the walls at top and bottom, the helicopter crashes and displays the score and ends the game.

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them. We would like to extend our sincere esteems towards our guide, **Mr. Shivraj B and Dr. Kiran Babu** for the support, guidance and encouragement, he provided during the BE Project. This work would have not been possible without his valuable time, patience and motivation. We thank him for making our stint thoroughly pleasant and enriching. We are deeply indebted to Prof. **Dr. Shreekanth Mooroor Prabhu** (Head of Department of CSE) and the entire team in the Computer Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work. We take the privilege to express our sincere thanks to **Dr. Sanjay Jain**, our Principal for providing the encouragement and much support throughout our work.

D Sai Kumar Reddy(1CR19CS035)

Darshan R (1CR19CS037)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

In this era of computing, computer graphics has become one of the most powerful and interesting fact of computing. It all started with display of data on hardcopy and CRT screen. Now computer graphics is about creation, retrieval, manipulation of models and images. Graphics today is used in many different areas. Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results. OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

1.2 OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross language cross platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex 3D scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation. It is also used in video games, where it competes with direct 3D on Microsoft Windows Platforms.
OpenGL serves two main purposes:

- To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API.

- To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

- Rasterized points, lines and polygons are basic primitives.
- A transform and lighting pipeline.
- Z buffering.
- Texture Mapping.
- Alpha Blending

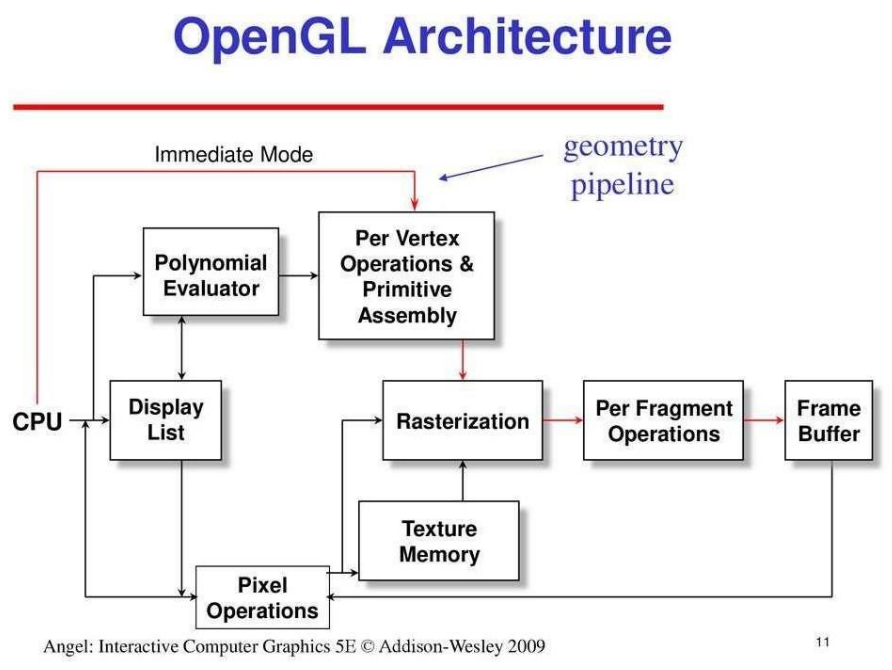


Fig 1.1 OpenGL Graphics Architecture

1.3 OpenGL Interface

Most of applications will be designed to access OpenGL directly through functions in three libraries. They are:

1. Main GL: Library has names that begin with the letter gl and are stored in a library usually referred to as GL.
2. OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.
3. OpenGL Utility Toolkit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system

It is stream lined in other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.

OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc. It is system-independent. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities. It is a state machine. At any moment during the execution of a program there is a current model transformation. It is a rendering pipeline. The rendering pipeline consists of the following steps: o Defines objects mathematically. Arranges objects in space relative to a viewpoint. o Calculates the color of the objects. Rasterizes the objects.

1.4 Project Goal

The aim of this project is to develop a 3D view which supports basic operations which include Movement, and transformation operations like translation, rotation, scaling etc on objects. The package must also have a user friendly interface.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 HARDWARE REQUIREMENT SPECIFICATIONS

a. Any Operating System

b. Any Computer Processors

c. Any screen resolution

2.2 SOFTWARE REQUIREMENT SPECIFICATIONS

This graphics package can be used in any platform.

a. Development Platform: WINDOWS

b. Development tool: CODEBLOCKS

c. Language Used In Coding: C

CHAPTER 3

DESIGN

3.1 Proposed System

To achieve three dimensional effects, OpenGL software is proposed. It is software which provides a graphical interface. It is an interface between application program and graphics hardware. The advantages are:

- OpenGL is designed as a streamlined.
- it is a hardware independent interface, it can be implemented on many different hardware platforms.
- With OpenGL, we can draw a small set of geometric primitives such as points, lines and polygons etc.
- It provides double buffering which is vital in providing transformations.
- It is event driven software.
- It provides call back function.

3.2 Transformation Functions

- Translation: Translation is done by adding the required amount of translation quantities to each of the points of the objects in the selected area. If P(x,y) be the a point and (tx, ty) translation quantities then the translated point is given by glTranslatef(dx,dy,dz);
- Rotation: The rotation of an object by an angle 'a' is accomplished by rotating each of the points of the object. The rotated points can be obtained using the OpenGL functions glRotatef(angle, vx,vy,vz);
- Scaling: The scaling operation on an object can be carried out for an object by multiplying each of the points (x,y,z) by the scaling factors sx, sy and sz. glScalef(sx,sy,sz);

CHAPTER 4

IMPLEMENTATION

4.1 DESCRIPTION:

GL primitives can have either flat or smooth shading. Smooth shading, the default, causes the computed colors of vertices to be interpolated as the primitive is rasterized typically assigning different colors to each resulting pixel fragment. Flat shading selects the computed color of just one vertex and assigns it to all the pixel fragments generated by rasterizing a single primitive.

4.2 FUNCTIONS IN OPEN-GL :

- **void glClear(glEnum mode);** Clears the buffers namely color buffer and depth buffer. mode refers to GL_COLOR_BUFFER_BIT or DEPTH_BUFFER_BIT.
- **void glTranslate[fd](TYPE x, TYPE y, TYPE z);** Alters the current matrix by displacement of (x, y, z), TYPE is either GLfloat or GLdouble.
- **void glutSwapBuffers();** Swaps the front and back buffers.
- **void glMatrixMode(GLenum mode);** Specifies which matrix will be affected by subsequent transformations, Mode can be GL_MODELVIEW or GL_PROJECTION.
- **void glLoadIdentity( );** Sets the current transformation matrix to identity matrix.
- **void glEnable(GLenum feature);** Enables an OpenGL feature. Feature can be GL_DEPTH_TEST (enables depth test for hidden surface removal), GL_LIGHTING (enables for lighting calculations), GL_LIGHTi (used to turn on the light for number i), etc.
- **void glPushMatrix();** void glPopMatrix(); Pushes to and pops from the matrix stack corresponding to the current matrix mode.
- **void glutBitmapCharacter(void *font, int character);** Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font. The fonts used are GLUT_BITMAP_TIMES_ROMAN_24, GLUT_BITMAP_ HELVETICA_18.
- **void glRasterPos[234][ifd](GLfloat x, GLfloat y, GLfloat z);** This position, called the raster position, is used to position pixel and bitmap write operations.

- **void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);** Defines an orthographic viewing volume with all the parameters measured from the centre of the projection plane.
- **void glutInit(int *argc, char **argv);** Initializes GLUT; the arguments from main are passed in and can be used by the application.
- **void glutInitDisplayMode(unsigned int mode);** Requests a display with the properties in the mode; the value of mode is determined by the logical OR of options including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE).
- **void glutCreateWindow(char *title);** Creates a window on display; the string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.
- **void glutDisplayFunc(void (*func)(void));** Registers the display function func that is executed when the window needs to be redrawn.
- **void glutMouseFunc(void *f(int button, int state, int x, int y);** Registers the mouse callback function f. The callback function returns the button (GLUT_LEFT_BUTTON,etc., the state of the button after the event (GLUT_DOWN), and the position of the mouse relative to the top-left corner of the window.
- **void glutKeyboardFunc(void *f(char key, int width, int height));** Registers the keyboard callback function f. The callback function returns the ASCII code of the key pressed and the position of the mouse. ▪
- **void glClearColor(GLclampf r, GLclampf g, GLclamp b, Glclamp a);** Sets the present RGBA clear color used when clearing the color buffer. Variables of type GLclampf are floating point numbers between 0.0 and 1.0.
- **void glViewport(int x ,int y, GLsizei width, GLsizei height);** Specifies the width*height viewport in pixels whose lower left corner is at (x,y) measured from the origin of the window.
- **void glColor3[b I f d ub us ui](TYPE r, TYPE g, TYPE b);** Sets the present RGB colors. Valid types are bytes(b), int(i), float(f), double(d), unsigned byte(ub), unsigned short(us), and unsigned int(ui). The maximum and minimum value for floating point types are 1.0 and 0.0 respectively, whereas the maximum and minimum values of the discrete types are those of the type, for eg, 255 and 0 for unsigned bytes.
- **void glutInitWindowSize(int width, int height);** Specifies the initial height and width of the window in pixels.

- **void glutReshapeFunc(void *f(int width, int height));** Registers the reshape callback function f. the callback function returns the height and width of the new window. The reshape callback invokes the display callback.
- **void createMenu(void);** This function is used to create menus which are used as options in program.

4.3 CODE:

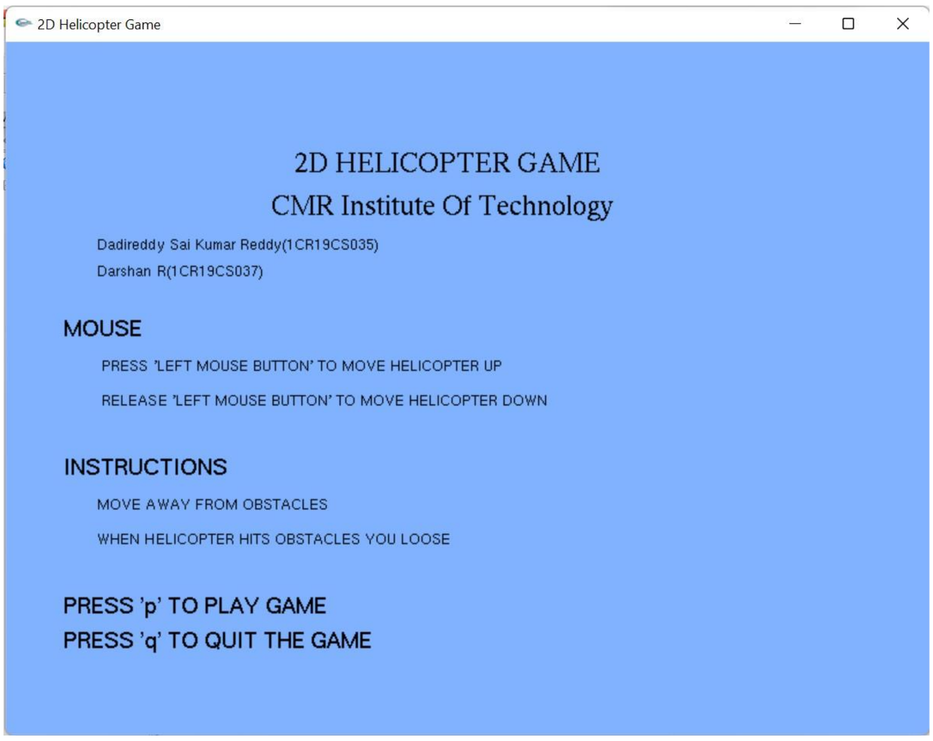
```
#define GLUT_DISABLE_ATEXIT_HACK
#include<stdlib.h>
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>
#include<stdio.h>
#include<iostream>
#include<windows.h>
#include<string.h>

float y=0;
float bx = 0,bx1=0;
float by  =  0,by1=0;
int view =0, score = 0;
float theta = 0.0;
char scoreText[10];
void display_string(int x, int y, char *string, int font)
{
  int len,i;
        glColor3f(0,0,0.0);
        glRasterPos2f(x, y);
  len = (int) strlen(string);
  for (i = 0; i < len; i++) {
  if(font==1)
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);
```

```
            if(font==2)
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,string[i]);
            if(font==3)
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12,string[i]);
            }
}
void gameStartDisplay(){
    glClearColor(0.5, 0.7, 1, 1);
    glClear(GL_COLOR_BUFFER_BIT);
            display_string(250, 650, "2D HELICOPTER GAME",1);
            display_string(230, 600, "CMR Institute Of Technology",1);
            display_string(80, 560, "Dadireddy Sai Kumar Reddy(1CR19CS035)",3);
            display_string(80, 530, "Darshan R(1CR19CS037)",3);
            display_string(50, 560, "MOUSE",2);
            display_string(80, 520, " PRESS 'LEFT MOUSE BUTTON' TO MOVE HELICOPTER
UP",3);
            display_string(80, 480, " RELEASE 'LEFT MOUSE BUTTON' TO MOVE HELICOPTER
DOWN",3);
            display_string(50, 400, "INSTRUCTIONS",2);
            display_string(80, 360, "MOVE AWAY FROM OBSTACLES",3);
            display_string(80, 320, "WHEN HELICOPTER HITS OBSTACLES YOU LOOSE",3);
            display_string(50, 240, "PRESS 'p' TO PLAY GAME",2);
            display_string(50, 200, "PRESS 'q' TO QUIT THE GAME",2);
            glutPostRedisplay();
            glutSwapBuffers();
            glFlush();
}
void helicopter()
{
    glColor3f(0, 0, 0);
    glRectf(100,150+y,200,100+y);        //body
    glRectf(145,150+y,155,170+y);        //top

    glRectf(60,115+y,100,130+y);         //tail1
```

```
        glRectf(54,115+y,60,160+y);          //t2

        glColor3f(0.5, 1, 1);
        glRectf(120,110+y,160,140+y);        //win1
        glRectf(170,120+y,190,140+y);        //win2
        glBegin(GL_TRIANGLES);
            glVertex2i(47,150+y);        //propeller
            glVertex2i(47,160+y);
            glVertex2i(57,155+y);
            glVertex2i(57,155+y);
            glVertex2i(67,150+y);
            glVertex2i(67,160+y);
        glEnd();
        glutPostRedisplay();
        glutSwapBuffers();
    }
    void moveBlock1(){
        bx -=0.1;
        if (bx < -1050)
        {
            bx = 0;            //total width is 50
            by = (rand() % 200) +50;
        }
        if  (bx < -700 && bx > -701)
        {
             score++;
        }
    }
    void moveBlock2(){
        bx1 -=0.1;
        if (bx1 < -1050)
        {
            bx1 = 0;            //total width is 50
            by1 = (rand() % 200)+0;
        }
        if  (bx1 < -1000 && bx1 > -1001)
        {
             score++;
        }

    }
    void gameOverDisplay(){
        glClearColor(0, 0, 0, 1);
        glClear(GL_COLOR_BUFFER_BIT);
        display_string(400, 700, "GAME OVER", 1);
        display_string(420, 650, "SCORE : ", 2);
        display_string(350, 550, "PRESS 'R' TO RETURN", 2);
        glFlush();
        printf("\n\n\t\t\t\t---------------THANKS -------------- \n");
        printf("\t\t\t\t........................FOR........................\n");
        printf("\t\t\t\t---------------PLAYING------------- \n\n\n\n");
        printf("\t\t\t\t\tScore:%s\n\n\n",scoreText);
        exit(0);
    }
    bool hit(float xb1, float yb1, float xb2, float yb2, float xh1, float yh1, float xh2, float yh2){
        if(xb1 >= xh1 && xb1 <=xh2 && yb1 <= yh1 && yb1 >= yh2){
            return true;
        }
        if(xb2 >= xh1 && xb2 <=xh2 && yb2 <= yh1 && yb2 >= yh2){
            return true;
        }
        return false;
    }

    void blocks1 (){
          glColor3f(1, 0.2, 0.2);
         // by= (rand() % 2000) ;
         glRectf(630 + bx,300+by1,650 + bx,550+by1);
         if(hit(630 + bx, 550 + by1, 650 + bx, 300 + by1, 60, 185 + y, 200, 100 + y)){
                gameOverDisplay();
         }
    }
    void blocks2 () {
        glColor3f(1, 0.2, 0.2);
        glRectf(1050 + bx1,by,1070 + bx1,250+by);
        if(hit(1050 + bx1, by, 1070 + bx1, 250 + by, 60, 185 + y, 200, 100 + y)){
                gameOverDisplay();
         }
    }
    void walls(){
        glColor3f(1,1,0);
        glRectf(0, 800, 800, 780);
        glRectf(0, 0, 800, 20);
    }
    void fans (){
         glColor3f(1, 1, 1);
        // glRotatef( 90, 0,1, 0 );
        glBegin(GL_POLYGON);
            glVertex2i(90,160+y);          //fan
            glVertex2i(210,180+y);
            glVertex2i(210,185+y);
            glVertex2i(90,165+y);
        glEnd();
        glBegin(GL_POLYGON);
            glVertex2i(90,180+y);          //fan
            glVertex2i(210,160+y);
            glVertex2i(210,165+y);
            glVertex2i(90,185+y);
        glEnd();
    }
    void display()
    {
        if(view == 0){
            gameStartDisplay();
        }
        else{
            itoa(score, scoreText, 10);
            glClearColor(0.7, 0.7, 1, 1);
            glClear(GL_COLOR_BUFFER_BIT);
            display_string(20, 750, "SCORE : ", 2);
            display_string(120, 750, scoreText, 2);
            helicopter();
            blocks1();
            blocks2();
            fans();
            walls();
            glFlush();
            glutPostRedisplay();
            glutSwapBuffers();
        }
    }
    void init()
    {
        glMatrixMode(GL_PROJECTION);
        gluOrtho2D(0, 800, 0, 800);
        glMatrixMode(GL_MODELVIEW);
    }
    void movehelUp(){
        /*if(key==GLUT_KEY_UP){
            if(y<600){
                y+=8;
            }
        }*/
        if(y>630){
            gameOverDisplay();
        }
        else{
            y+=0.15;
        }
        moveBlock1();
        moveBlock2();
    }
    void movehelDown(){
         if(y<-100){
            gameOverDisplay();
        }
        else{
            y-=0.2;
        }
        moveBlock1();
        moveBlock2();
        glutPostRedisplay();
    }
    void mouse(int button, int state, int x, int y)
    {
                switch (button)
                {
                case GLUT_LEFT_BUTTON:

                        if (state == GLUT_DOWN){
                                glutIdleFunc(movehelUp);
                        }
                        else if (state == GLUT_UP){
                                glutIdleFunc(movehelDown);
                        }
                        break;
                default: break;
                }
    }
    void keyboard(unsigned char key,int u,int d){
        switch(key){
            case 'p': view = 1;
            break;
            case 'r': if(view != 0){
                    view = 0;
                }
            break;
            case 'q' :exit(0);

        }
    }
    int main(int argc, char** argv)
    {
        glutInit(&argc, argv);
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize (800, 600);
        glutInitWindowPosition (0,0);
        glutCreateWindow ("2D Helicopter Game");
        init();
        glutDisplayFunc(display);
        glutMouseFunc(mouse);
        glutKeyboardFunc(keyboard); //keyboard call back func
        glutMainLoop();
        return 0;
    }
```

CHAPTER 5

SCREENSHOTS



2D Helicopter Game

2D HELICOPTER GAME

CMR Institute Of Technology

Dadireddy Sai Kumar Reddy(1CR19CS035)

Darshan R(1CR19CS037)

MOUSE

PRESS 'LEFT MOUSE BUTTON' TO MOVE HELICOPTER UP

RELEASE 'LEFT MOUSE BUTTON' TO MOVE HELICOPTER DOWN

INSTRUCTIONS

MOVE AWAY FROM OBSTACLES

WHEN HELICOPTER HITS OBSTACLES YOU LOOSE

PRESS 'p' TO PLAY GAME

PRESS 'q' TO QUIT THE GAME

Fig 5.1 : Game Instructions Window

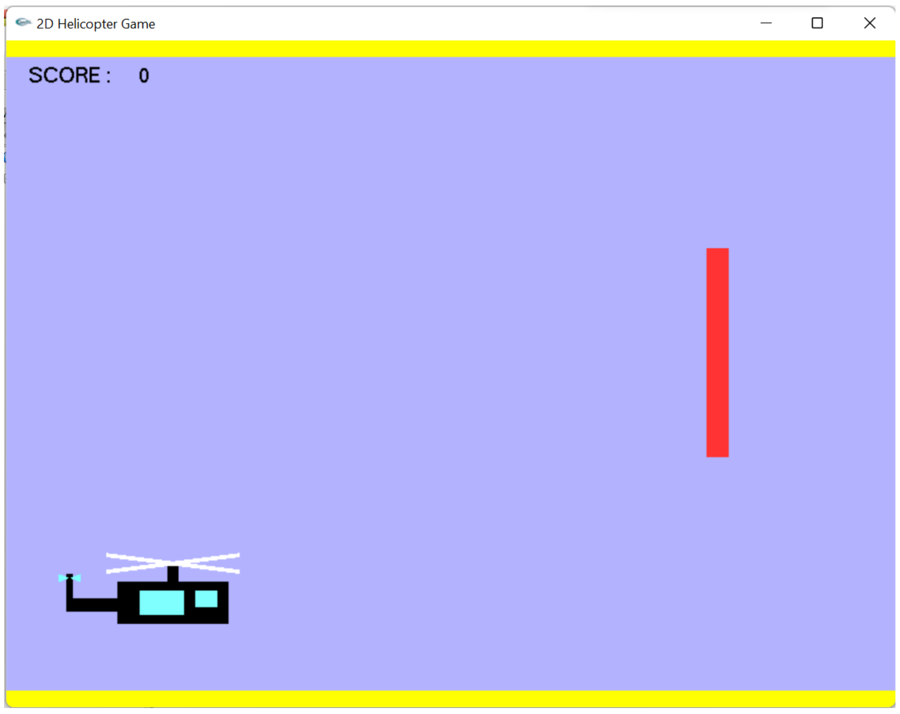


Fig 5.2 : Game Start Window

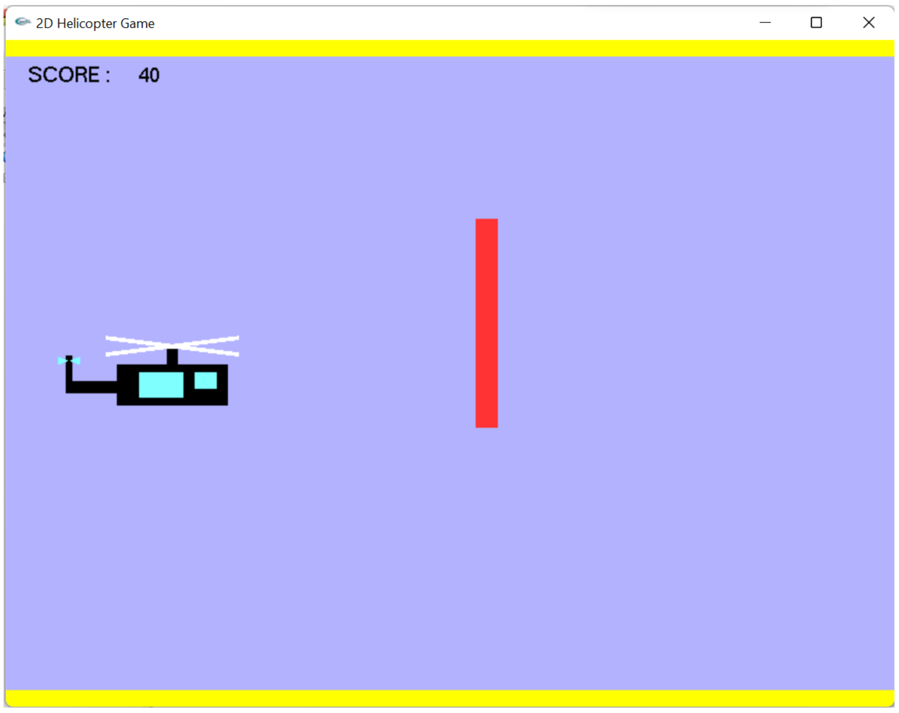


Fig 5.3 : In Game Window

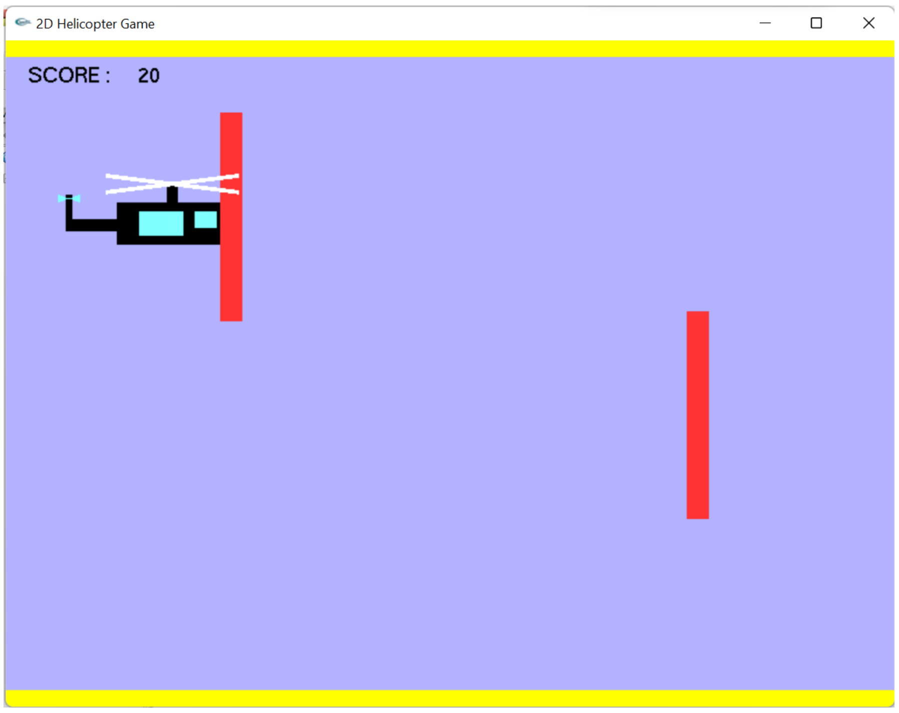
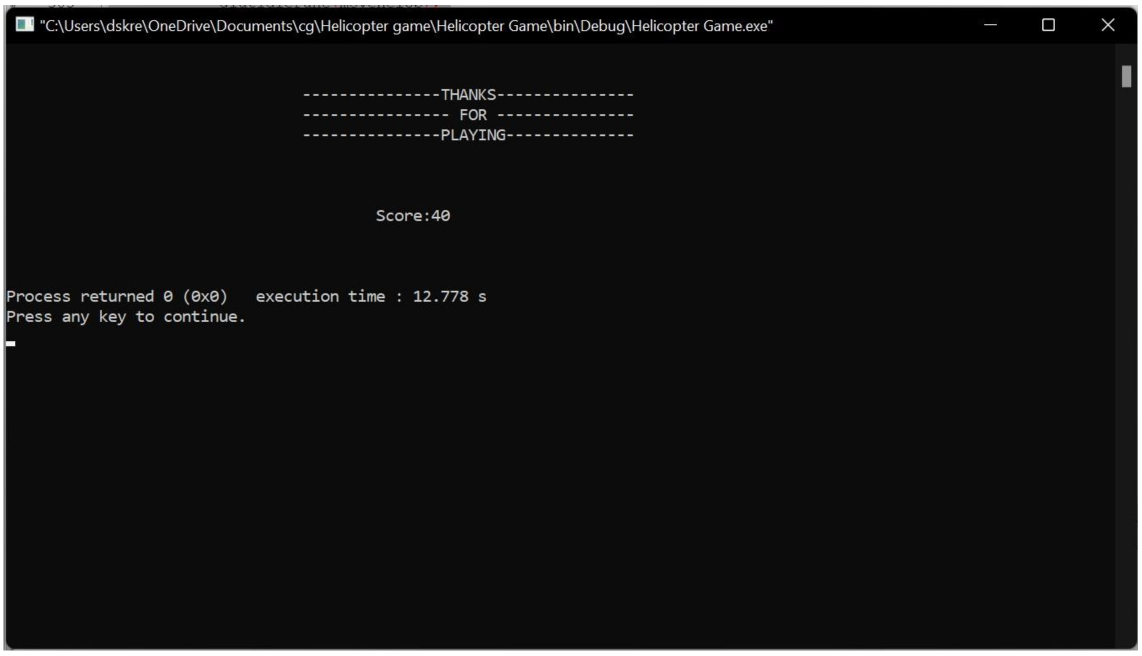


Fig 5.4 : Helicopter Crash Window



"C:\Users\dskre\OneDrive\Documents\cg\Helicopter game\Helicopter Game\bin\Debug\Helicopter Game.exe"

```
                                ---------------THANKS---------------
                                ---------------- FOR ---------------
                                ---------------PLAYING--------------



                                        Score:40



Process returned 0 (0x0)   execution time : 12.778 s
Press any key to continue.
```

Fig 5.5 : End Game and Results Window

CHAPTER 6

CONCLUSION & FUTURE SCOPE

A 2d graphics based game helicopter is a great start for a student who starts learning computer graphics & visualization. The development of the game has large scope to learn computer graphics from scratch. We used OpenGL utility toolkit to implement the algorithm, written it in c++ language.

There is still scope left in the development of project like, after "game over" a list should show top ten scorers, a need to embed a button "play again". Welcome screen need more modification there is scope of embedding buttons like "about", "how to play", "configuration", "profiles", etc. In future we hope we would implement it in source code for better experience of playing this game. Finally, we could say by developing the game we have learnt the basics of f computer graphics and in future by developing it further we shall learn more. It will be our pleasure if we could develop in 3d graphics package.

BIBLIOGRAPHY

[1] http://vtucs.com/computer-graphics-projects/

[2] https://www.youtube.com/watch?v=45MIykWJ-C4

[3] Getting started with OpenGL - GeeksforGeeks